

**電子工程系碩士班**

**碩士學位論文**

**Department of Electronic Engineering**

**Master Thesis**

**超大型數據集的高效用模式探勘近似隨機演算法**

**A Probabilistic Approximate High-utility Pattern Mining Algorithm for Ultra Large Scale Datasets**

**研究生：呂彥旻**

**指導教授：黃柏鈞 博士**

**Advisor: Po-Chun Huang, Ph.D.**

**August 2023**

摘 要

論文名稱：超大型數據集的高效用模式探勘近似隨機演算法

頁數：頁

校所別：國立臺北科技大學 電子工程系 碩士班

畢業時間：一百一十一學年度 第二學期

學位：碩士

研究生：呂彥旻

指導教授：黃柏鈞 博士

關鍵詞：高效用模式探勘、隨機演算法、近似演算法.

在一個潛在龐大的資料集中，探索重要的模式通常具有相當大的價值。此類問題中最基本的形式是頻繁模式的挖掘，它的目的是發現在資料集中出現次數最多的項目集合。雖然已經有很多演算法來解決頻繁模式挖掘的問題，但是現實的環境通常更加複雜。例如：假設在數據集中，每一個項目都可能具有不同的價值，因此有了高效用模式挖掘，目的在於發現數據集中具有最高效用的項目集合。由於高效用模式不具有反單調的特性，因此高效用模式的子集不一定保證也是高效用模式。而缺乏反單調性特性，使得在龐大的數據集中挖掘高效用模式比頻繁模式更加困難。

在這項研究中，我們觀察到在現實生活中通常不需要進行確定性和精確性的高效用模式挖掘。相反，我們通常只需發現一些具有高概率是高效用模式的項目集，而不一定是真正具有最高效用的項目集。透過放寬這些要求，我們提出一種近似高效用模式挖掘演算法（PAHUPMA），他可以在給定的數據集中發現高效用模式，而無須掃描整個數據集。因此，高效用模式挖掘所產生的 IO 流量大幅減少，而顯著提升挖掘的性能。根據我們初步的實驗結果，PAHUPMA 所花費的只是現有演算法（如 HUI-Miner 和 HUPM）的一小部分。我們相信，PAHUPMA的優異性能能夠有效擴展高效用模式挖掘的應用範疇。

ABSTRACT

Title: A Probabilistic Approximate High-utility Pattern Mining Algorithm for Ultra Large Scale Datasets

Pages:

School: National Taipei University of Technology

Department: Department of Electronic Engineering

Date: August, 2023

Degree: Master

Researcher: Yan-Min Lu

Advisor: Po-Chun Huang, Ph.D.

Keywords: High-utility pattern mining, probabilistic algorithms, approximate algorithms.

In a potentially huge dataset, it is often useful to discover the important patterns. The simplest form of such problems is the frequent pattern mining, which asks to discover the itemsets that appear for most times in the dataset. While there have been quite some efficient algorithms for solving the frequent pattern mining problem, more sophisticated forms of the problems are often of greater interest. For example, given the utility of each item in the dataset, the high-utility pattern mining problem asks to discover the itemsets with the highest total utility in the dataset. As high-utility patterns do not exhibit the anti-monotonicity property, with which a subset of a highutility pattern is guaranteed to be a high-utility pattern as well. Without the anti-monotonicity, the mining of high-utility patterns in a huge dataset becomes much more difficult than that of frequent patterns.

In this work, we observed that the deterministic and exact high-utility pattern mining is often unnecessary in practice. Instead, it often suffices to discover some, but not all, patterns that have a high, but might not 100%, probability to have a high, but possibly not the highest total utility, in the given dataset. By relaxing such requirements, we present a probabilistic approximate high-utility pattern mining algorithm (PAHUPMA), which can discover the high-utility patterns in a given dataset without scanning through the whole dataset. As a result, the IO traffic generated for high-utility pattern mining is significantly reduced, which remarkably enhance the mining performance. According to our preliminary experimental results, the latency of PAHUPMA is only remnants of that of the existing deterministic algorithms, such as HUI-Miner, EFIM, HMiner, and ULB-Miner. We believe that the outstanding performance of PAHUPMA can effectively extend the spectrum of applications of high-utility pattern mining.

Table of Contents

[摘 要 i](#_Toc139826676)

[ABSTRACT iii](#_Toc139826677)

[Acknowledgements v](#_Toc139826678)

[Table of Contents vi](#_Toc139826679)

[List of Tables vii](#_Toc139826680)

[List of Figures viii](#_Toc139826681)

[Chapter 1 INTRODUCTION 1](#_Toc139826682)

[Chapter 2 BACKGROUNDS AND MOTIVATIONS 4](#_Toc139826683)

[A. System Architecture 4](#_Toc139826684)

[B. Pattern Mining in a Dataset 5](#_Toc139826685)

[C. Motivations 7](#_Toc139826686)

[Chapter 3 A PROBABILISTIC APPROXIMATE HIGH-UTILITY PATTERN MINING ALGORITHM (PAHUPMA) 8](#_Toc139826687)

[A. The Main Workflow 8](#_Toc139826688)

[Chapter 4 EXPERIMENTAL STUDIES 11](#_Toc139826689)

[A. Experimental Setting 11](#_Toc139826690)

[B. Experimental Results 11](#_Toc139826691)

[Chapter 5 RELATED WORK 13](#_Toc139826692)

[A. Frequent Pattern Mining (FPM) Algorithms 13](#_Toc139826693)

[B. High-utility Pattern Mining (HUPM) Algorithms 14](#_Toc139826694)

[Chapter 6 CONCLUSION AND FUTURE WORK 16](#_Toc139826695)

[REFERENCES 18](#_Toc139826696)

List of Tables

[Table 5.1 Environmental settings 25](#_Toc112370953)

List of Figures

[Figure 2.1 System architecture 9](#_Toc112373290)

[Figure 3.1 System Architecture. 12](#_Toc112373291)

[Figure 4.1 Zones in an existing ZNS SSD. 15](#_Toc112373292)

[Figure 4.2 Elastic zones in our proposed ZNS SSDs. Sequential zone and the parasite zones of the same texture indicate that they belong to the same elastic zone. 16](#_Toc112373293)

[Figure 4.3 Parasite zones mapping. The example of parasite zone mapping on ZNS SSDs. 16](#_Toc112373294)

[Figure 5.1 Zone switching. 25](#_Toc112373295)

[Figure 5.2 Zone flush. 26](#_Toc112373296)

# INTRODUCTION

Recently, thanks to the rapid development of data mining algorithms, the value of data has been widely recognized in diversified application scenarios, such as smart manufacturing, logistics, healthcare, banking, etc. Among the popular data mining technologies, it is common to discover the important patterns or itemsets in a given dataset, so as to unveil the hidden trends. According to the different ways to define “important” patterns in different application scenarios, such pattern mining problems may be further classified into several flavors. The simplest and probably the most representative flavor of pattern mining problems is the frequent pattern mining (FPM) problems, which ask to discover the most frequently occurring patterns, i.e., those with the highest count of occurrences, in a given dataset. Classically, if there exist multiple patterns with the largest and the same counts of occurrences in the dataset, all of them should be discovered and returned. Existing algorithms to the FPM problem often rely on the anti-monotonicity of frequent patterns, i.e., all subsets of frequent patterns are also frequent patterns, to accelerate the mining of frequent patterns.

In practice, it is often the case that some data items in the dataset are more important than the others. In this case, each data item may be associated with a weight or utility, where a larger utility indicates higher importance. Instead of finding the patterns with the highest occurrence count, such a highutility pattern mining (HUPM) problem asks to discover all patterns with the highest utility sum of all data items in the patterns in the dataset. Although the HUPM problem exhibits a similar structure to the FPM problem does, the HUPM problem is much more challenging than the FPM problem, due to the absence of anti-monotonicity of high-utility patterns. As a result, existing HUPM algorithms often need to scan the dataset for multiple passes or maintain large intermediate data structures of partially mining results or metadata, which significantly amplifies the read traffic and slows down the mining performance.

To improve the performance of HUPM algorithms, it is crucial to reduce the IO traffic of the secondary storage, which is typically slower by several orders of magnitude than the SRAM or DRAM based main memory. However, due to the aforementioned reasons, existing HUPM algorithms often fail to achieve the design objective, motivating this work. Fortunately, in many application scenarios it is often unnecessary to discover all patterns with the highest utility among all possible patterns of the dataset. Instead, finding a pattern with sufficiently high utility in a much shorter time is often useful enough. Going a step further, having a sufficiently high probability to discover some, but not all, patterns of sufficiently high, but not the highest, utilities is often useful enough for some specific application scenarios. This inspires the adoption of some probabilistic and approximate algorithms that strive to skip unimportant transactions in the dataset by not loading them into the main memory for analysis.

In this work, we propose a novel algorithm called the probabilistic approximate high-utility pattern mining algorithm (PAHUPMA) for high-utility pattern mining. The basic rationale behind PAHUPMA is to avoid loading a considerable part of transactions in the dataset from the secondary storage without seriously degrading the quality (i.e., the number and utility) of the discovered patterns. As the utility of a pattern in a dataset is defined by the sum of the utility of all data items in the pattern timed by the occurrence count of the pattern in the dataset, the PAHUPMA groups multiple transactions in the dataset and stored them together in the same page sets of the storage. (The unit “page” is also termed as a sector and page for mechanical hard disk and NAND flash memory, respectively.) In addition, the sum of the utility of all transactions in the same group will be maintained in the main memory or some dedicated storage space. During the high-utility mining process, a group with higher utility will be granted a larger probability to be loaded into the limited main memory for analysis using the existing HUPM algorithms, such as HUI-Miner [1] or EFIM [2]. Thus, from the long-term perspective, patterns with higher actual utility in the dataset have larger expected utility. This is due to that, a pattern with a large sum of the utilities of its data items will have a high probability to be loaded on every occurrence in the dataset. On the other hand, a pattern with a large occurrence count in the dataset will be granted more chances to be loaded into the main memory and considered in the mining process. For a pattern with the highest utility among all patterns in the dataset, it will have the highest expected utility in the main memory, and thus discovered by the existing HUPM algorithm in the second phase. With the PAHUPMA, the average amount of transaction data that has to be loaded from the secondary storage into the main memory could be much smaller than the whole dataset. As a result, the IO overheads can be remarkably suppressed at reasonable loss in the quality of the discovered patterns.

The rest of this thesis is organized as follows. First of all, Section II presents the target system architecture, defines the high-utility pattern mining problem, and then motivates this work. After that, Section III presents the probabilistic approximate high-utility pattern mining algorithm (PAHUPMA), an approximate HUPM algorithm based on dice-rolling. To evaluate the efficacy of PAHUPMA, a series of experimental studies are performed, with their results reported and discussed in Section IV. The important prior literature on frequent pattern mining and high-utility pattern mining are discussed as in Section V, followed by Section VI that concludes the thesis and outlines potential future research directions.

.

# BACKGROUNDS AND MOTIVATIONS

## System Architecture

In this section, we present the system architecture adopted by PAHUPMA, as well as many existing high-utility pattern mining schemes, as in Figure 1. In this work, we assumed that there is a sufficiently large persistent storage, which stores the whole dataset as well as the associated metadata. On the other hand, there is a main memory space that is much smaller, say 1%, yet much faster, say 10x in terms of throughput, as compared to the storage. The main memory can be leveraged to maintain the temporary data required by high-utility mining algorithms, such as the candidates of highutility patterns and their utilities. The main memory is byte addressable and accessible; however, the storage has to be accessed in the coarse-grained unit of blocks, which is also called sectors on mechanical hard disks and pages on NAND flash memory solid-state disks (SSDs).

Concerning the big gaps of access performance and storage capacity of the main memory and storage, high-utility pattern mining algorithms must eliminate unnecessary accesses to the storage to improve the mining performance. On the other hand, some high-utility pattern mining algorithms tend to generate a large amount of temporary data, such as the candidates of highutility patterns, in the main memory. However, if the temporary data cannot be accommodated in the main memory, those data have to be stored on the storage instead and managed in an ondemand manner, which further amplifies the mining overheads. How to reduce the main memory usage while suppressing unnecessary storage accesses therefore becomes an important issue.

## Pattern Mining in a Dataset

Pattern mining is a significant technical problem that discovers valuable hidden trends behind a potentially large volume of data. Usually, there would be a number of transactions in a dataset, where each transaction has a corresponding itemset. Each itemset comprises one or more data items. Depending on the different flavors of pattern mining problems, each distinct data item may be limited to appear for only once or multiple times in a itemset, and possibly associated with a nonnegative weight or utility that denotes the relative significance of the item. Although there might be various ways to identify different data items, it can be assumed that there is an integer index number (id) associated with each distinct data item for the simplicity of discussion. A pattern is a set of data items, with its support counter denoting the number of occurrences of the pattern in the dataset. In a pattern mining problem, given a dataset, possibly with the weight of each data item, we are asked to discover the “important” patterns in the dataset.

The simplest form of pattern mining problems might be the frequent pattern mining (FPM) problem. In the FPM problem, we are asked to find the most frequently-occurring patterns— those with the largest support counter—in the dataset. Note that, due to the anti-monotonicity of frequent patterns, all subsets of a frequent pattern must also be frequent patterns, because the support counter of these subsets must be no smaller than that of the frequent pattern. There have been a number of classical approaches to the FPM problem, such as the Apriori algorithm that maintains current candidates of frequent patterns [3] and the frequent-pattern tree (FPtree) that leverages a prefix tree structure to reduce the main memory usage [4].

As a straightforward extension of the FPM problem, the high-utility pattern mining (HUPM) allows to assign a weight for every data item, and asks to discover the patterns with the highest utility sum in the given dataset. When the utility of every data item are the same, the HUPM problem degenerates to the FPM problem. However, the general HUPM problem is harder than the FPM problem because of the absence of the anti-monotonicity. For example, let us consider a minimum dataset with only two transactions T1: ⟨⟩ and T2: ⟨, ⟩. The utilities of the two items and in the dataset are both 1. In the dataset, although the pattern ⟨, ⟩ has the highest utility of 2, however, only its proper sub-pattern ⟨⟩ has the same highest utility of 2 and is also a high-utility pattern, while the other proper sub-pattern ⟨⟩ has a lower utility of 1 and is not a high-utility pattern. Due to the absence of antimonotonicity, the HUPM problem is much harder than the FPM problem. Specifically, the classical FP-tree [4] cannot be directly applied to solve the HUPM problem, because the prefix of a high-utility pattern—which represents a proper subpattern—does not necessarily represent a high-utility pattern.

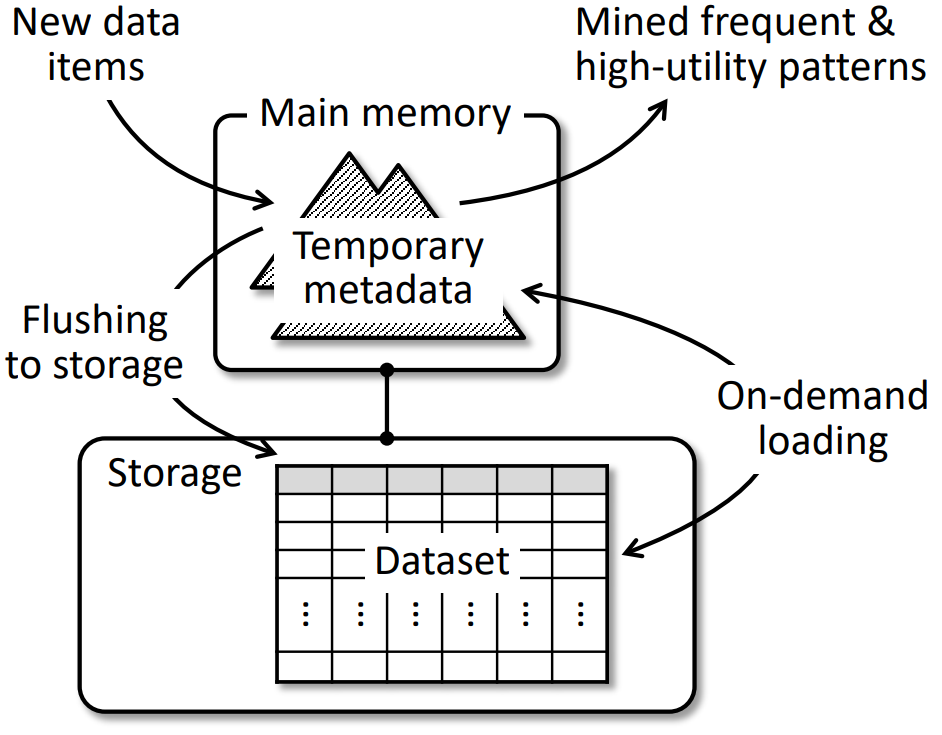


Figure . System architecture.

## Motivations

There have been numerous algorithms of pattern mining [5], [6] and HUPM [7]–[10]. In many HUPM algorithms, the major performance overheads originates from the IO overheads to load the potentially large amount of data in the dataset, and to maintain the temporary data required by the HUPM algorithms that cannot be accommodated in the main memory. Thus, (1) how to reduce the number of passes that the HUPM algorithms need to scan over the dataset, and (2) given the size of the main memory, how to control the amount of temporary data of the HUPM algorithms with the main memory size, have become design highlights in the prior arts. Notably, such a design challenge would exacerbate as the datasets are expected to become larger in the future, and must be alleviated.

In many application scenarios, it is often unnecessary to solve the original version of HUPM problem. For example, let us consider a very large database of transaction records collected from a supermarket, where both the numbers of goods and transactions are large. The utilities of patterns then denote the potential revenues of the corresponding goods. To optimize the promotion strategy of goods, it is unnecessary to discover all high-utility patterns; finding some pattern with the highest utility is often enough. Going a step further, sometimes it is unnecessary to find a pattern with the highest utility; finding a pattern that has a sufficiently high utility is often enough. To summarize, as compared to the existing HUPM algorithms that repetitively scan the dataset and discover all high-utility patterns, sometimes a HUPM algorithm that does not need to scan the whole dataset and discover only some high-utility patterns might be more useful, especially when the size of the dataset is extremely large. Based on the key observation, we present the probabilistic approximate high-utility pattern mining algorithm (PAHUPMA), a HUPM algorithm that trades mining quality for significantly enhanced mining performance.

# A PROBABILISTIC APPROXIMATE HIGH-UTILITY PATTERN MINING ALGORITHM (PAHUPMA)

## The Main Workflow

Although with a similar basic problem model, high-utility pattern mining is much more difficult than frequent pattern mining, due to the lack of anti-monotonicity property [11], [12]. As a result, prior work tend to scan through the whole and possibly huge datasets kept on secondary storage devices instead of the main memory. Concerning the much lower performance of storage devices than the main memory, it is crucial to reduce the IO traffic generated for high-utility pattern mining. To pursue this goal, we propose a probabilistic approximate high-utility pattern mining algorithm (PAHUPMA), which only needs to scan a small fraction of the whole dataset to discover high-utility patterns (Algorithm 1).

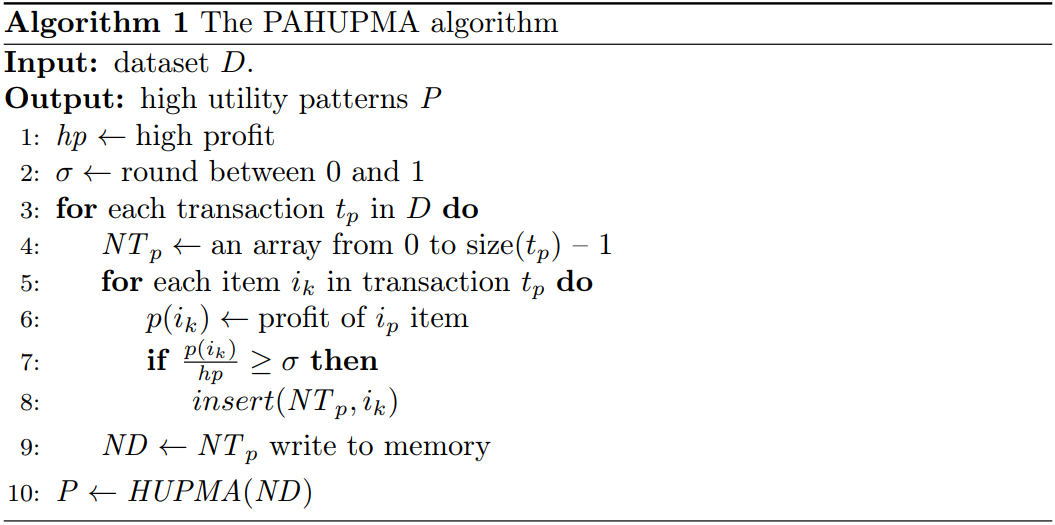
The main idea of our proposed algorithm is that, a transaction with a higher total utility is granted a higher probability to be loaded from the storage and considered in the discovery of high-utility patterns. As a result, many transactions with low total utilities can be skipped without significantly affecting the discovered high-utility patterns. However, there might be a very large number of transactions in the considered dataset, making it impossible to keep the total utilization of every transaction in the main memory. In PAHUPMA, we propose a transaction grouping technique which organizes every fixed number of transactions in a transaction group, and maintain the total utility of all items in all transactions of the group in main memory. We then consider the total utility of each transaction group to probabilistically determine whether the whole group can be skipped in the discovery of high-utility patterns. As a result, the IO traffic generated to load the transaction groups for the discovery of high-utility patterns is dramatically reduced.

Whether a transaction group should be considered or not is determined by the relative utility of the group. Specifically, as each group is inserted into the dataset, the maximum utility among the utilities of all transaction groups is maintained. A group with utility of is then granted a probability of to be considered, where. Thus, in the extreme case, a group with the maximum utility is guaranteed to be considered, while a group whose utility is very low against the maximum utility is granted a very low probability to be considered. The rationale behind the equation of originates from the definition of the utilities of patterns, which are equal to the product of the utilities of the items in the patterns and the frequencies of occurrence of the patterns. For a pattern with very high item utilities, its residing transaction groups will have a very high probability to be considered. On the contrary, a frequent occurring pattern whose item utilities are small will have many chances to be examined in each corresponding transaction group, and is also likely to be considered in the mining process of high-utility patterns

As PAHUPMA goes through the whole dataset, it either skips or loads then considered each transaction group for the discovery of high-utility patterns. At this time, any existing approaches to high-utility pattern mining may be used, as if there were only the loaded transaction groups in the dataset, while the skipped transaction groups had never existed. For example, all the patterns in the loaded transaction groups are maintained in some data structure such as a prefix tree [13] along with their utilities. After all transaction groups have been either skipped or considered, the data structure will be used to determine the high-utility patterns in the whole dataset. How many and what patterns should be returned as the mining result at this time are determined by the nature of the criteria of the specific application scenarios.

Classically, high-utility pattern mining is to discover a fixed number of patterns with the highest utilities. Along with the high-utility patterns themselves, the utilities of these patterns are also returned. The approximate version of the problem then finds the same number of patterns that are expected to be of high utilities. However, it is impossible for our approach, PAHUPMA, to return the precise value of the discovered high-utility patterns, as only a small part of the transaction groups and transactions in the dataset has been considered. Fortunately, PAHUPMA can estimate the utility of each discovered pattern in the original whole dataset from the that in the considered part. Specifically, suppose that the utility of some discovered pattern p in the considered part is , then the actual utility of can be estimated as , where and are the total utility of the whole dataset and that of the considered part, respectively. Note that and can be easily obtained during the mining process of high-utility patterns.

Algorithm 3.1 PAHUPMA



# EXPERIMENTAL STUDIES

## Experimental Setting

In

On

The

## Experimental Results

To .

an

Wh

.

# RELATED WORK

## Frequent Pattern Mining (FPM) Algorithms

As the simplest form of mining problems for important patterns in a potentially large dataset, there have been numerous brilliant algorithms to the frequent pattern mining (FPM) problem [3], [13], [17], [18]. An early solution to the FPM problem is the Apriori algorithm, which generates and incrementally maintains a list of candidates of frequent patterns [3]. Unfortunately, due to the well-known anti-monotonicity of frequent patterns, many different frequent patterns might share the same data items. Consequently, the explicit maintenance of all frequent pattern candidates in the main memory would be space inefficient, because some data items might repetitively occur in different candidate patterns. Thus, to improve the space efficiency of the Apriori algorithm, some efforts have been dedicated to optimize the in-memory data structure and suppress redundant memory usage, such as the wellknown frequent-pattern growth (FP-Growth) algorithms and the accompanying frequent-pattern tree (FP-tree) [4]. With its prefix-tree like structure, the FP-tree space economically stores the most frequently occurring data items in the shallowest nodes of the FP-tree—those closest to the root—and lets different frequent pattern candidates share the same frequent data items to reduce the consumption of the main memory space.

To further improve the performance and energy efficiency of FP-Growth and FP-tree on modern persistent memories (PMs) such as phase-change memories (PCMs), some subsequent results have been reported. In particular, the evergreen FP-tree (EvFP-tree) [19] presents a lazy counter in the FP-tree to eliminate redundant updates of support counters and significantly reduce the write traffic to the PMs. Moreover, concerning the limited write endurance of PM cells, EvFP-tree also presents a minimum-bit-altered (MBA) encoding scheme so that different bits of a counter can be equally and minimally updated when the counter increases from zero to its maximum value. Based on EvFP-tree, another work called parallel EvFP-tree (PevFPtree) [20] has been proposed to exploit the inherent parallelism of PM devices and boost the performance of frequent pattern mining..

The application of the tree compression idea to the PevFPtree, the CpevFP-tree [21], is proposed to omit the creation of unimportant nodes of the FP-tree, thereby reducing the space usage of the main memory and reduce the IO traffic due to the on-demand loading caused by the insufficient main memory space. Besides, the CpevFP-tree also suggests a link merge algorithm so that multiple local FP-trees mined by individual processor cores or cluster nodes may be efficiently merged into one global FP-tree, which provides the final mining results. The CpevFP-tree also suggests a hash walk algorithm to improve the performance to walk an FP-tree or compressed FP-tree, especially when there could be a very large number of distinct data items in the dataset. When the datasets outscale and contain more data items, we believe that there are still plenty of research opportunities for further improving the design of FP-tree and its variants, so as to achieve even higher scalability, performance, space efficiency, and energy efficiency.

## High-utility Pattern Mining (HUPM) Algorithms

To As compared to the FPM problem, the problem of highutility pattern mining (HUPM) is more difficult, due to the absence of anti-monotonicity in high-utility patterns. Nevertheless, it is still possible to first discover a maximal itemset of a dataset, which is a high-utility pattern with the most data items in it [22], [23]. Afterwards, the mining algorithm will recognize all proper sub-patterns of the maximal itemset are all high-utility ones, as if the anti-monotonicity were applied to the HUPM problem. This trades some degradation of the quality of the obtained patterns for better performance of HUPM process, as not all high-utility patterns have to be examined. While the maximal itemsets are helpful for improving the performance of high-utility pattern mining, however, the utility of the sub-patterns cannot be known, and some patterns with very low utility might be undesirably returned.

The second research highlight of HUPM problem strives to turn the offline HUPM algorithms—such as HUI-Miner [1] and EFIM [2]—into online ones so that the list of high-utility patterns may be incrementally updated as the dataset is being inserted with new transactions from time to time [?]. For instance, some work maintains the utility list structure, which is a reversed representation of transactions in the dataset, and incrementally maintains the list to continuously discover high-utility patterns as the dataset is being inserted with new transactions [1], [24]–[26]. In some application scenarios, the utility of some or all data items in the dataset might be zero or even negative [27], [28]. Thus, there are also proposals that leverage the remainder utility—which is calculated from the transaction-weighted utility of individual data items—in the utility list for handling negative utilities [27], [28].

# CONCLUSION AND FUTURE WORK

High-utility pattern mining (HUPM) is a highly valuable but challenging variant of pattern mining problems. To solve the HUPM problem, existing approaches often need to scan through the dataset for multiple times, or generate a large volume of temporary data. Consequently, they might generate heavy IO traffic to the storage devices, which dramatically degrades the performance of high-utility pattern mining. Fortunately, in this work, we observe that it is often unnecessary to discover all patterns with the highest utility in the dataset. Instead, it often suffices to discover sufficiently many patterns, each of which has a high probability to have a sufficiently high utility in the dataset.In the future, we shall explore the system-level solutions, such as file systems and databases, for ZNS SSDs with our proposed elastic zone support. Moreover, to more precisely evaluate the efficacy of the elastic zones, we also plan to reperform the experimental studies on realistic hardware of ZNS SSDs, which also opens the possibility of future development of ZNS SSD designs.

From the definition of the utility of a pattern, we propose the probabilistic approximate high-utility pattern mining algorithm (PAHUPMA), which is an approximate HUPM algorithm based on dice rolling. By averting scanning the target dataset for more than one time, PAHUPMA significantly reduces the IO traffic overheads of the mining process, thereby remarkably enhancing the mining performance. Unlike existing algorithms, the PAHUPMA is adaptive to the different ratios of the space of the main memory and that of the storage, and achieves a more flexible trade-off between the quality of mining result (i.e., the number and utility of the mined patterns) and the mining performance (i.e., the IO traffic generated during the mining process). Furthermore, PAHUPMA can seamlessly collaborate with many different HUPM algorithms. According to our preliminary experimental studies, the PAHUPMA shows its superiority against existing deterministic HUPM algorithms, especially from the view of algorithmic scalability.

REFERENCES

1. M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” pp. 55–64, 2012.
2. S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, “Efim: a fast and memory efficient algorithm for high-utility itemset mining,” Knowledge and Information Systems, vol. 51, no. 2, pp. 595– 625, 2017.
3. R. Agrawal, R. Srikant et al., “Fast algorithms for mining association rules,” in Proc. 20th int. conf. very large data bases, VLDB, vol. 1215. Santiago, Chile, 1994, pp. 487–499.
4. J. Han, J. Pei, Y. Yin, and R. Mao, “Mining frequent patterns without candidate generation: A frequent-pattern tree approach,” Data mining and knowledge discovery, vol. 8, pp. 53–87, 2004..
5. P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, “A survey of sequential pattern mining,” Data Science and Pattern Recognition, vol. 1, no. 1, pp. 54–77, 2017.
6. W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. S. Tseng, and S. Y. Philip, “A survey of utility-oriented pattern mining,” IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 4, pp. 1306–1327, 2019.
7. H. Yao, H. J. Hamilton, and L. Geng, “A unified framework for utilitybased measures for mining itemsets,” in Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining. Citeseer, 2006, pp. 28–37.
8. Y. Liu, W.-k. Liao, and A. Choudhary, “A two-phase algorithm for fast discovery of high utility itemsets,” in Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9. Springer, 2005, pp. 689– 695.
9. HM. M. Bala and R. Dandamudi, “Hupm: Efficient high utility pattern mining algorithm for e-business,” in 2018 IEEE 8th International Advance Computing Conference (IACC). IEEE, 2018, pp. 191–195.
10. P. Fournier-Viger, J. C.-W. Lin, R. Nkambou, B. Vo, and V. S. Tseng, “High-utility pattern mining,” Cham: Springer, 2019.
11. C. K.-S. Leung, “Mining uncertain data,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 4, pp. 316– 329, 2011.
12. R. T. Ng, L. V. Lakshmanan, J. Han, and A. Pang, “Exploratory mining and pruning optimizations of constrained associations rules,” ACM Sigmod Record, vol. 27, no. 2, pp. 13–24, 1998.
13. V. S. Tseng, B.-E. Shie, C.-W. Wu, and S. Y. Philip, “Efficient algorithms for mining high utility itemsets from transactional databases,” IEEE transactions on knowledge and data engineering, vol. 25, no. 8, pp. 1772–1786, 2012.
14. P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, V. S. Tseng et al., “Spmf: a java open-source pattern mining library.” J. Mach. Learn. Res., vol. 15, no. 1, pp. 3389–3393, 2014.
15. S. Krishnamoorthy, “Hminer: Efficiently mining high utility itemsets,” Expert Systems with Applications, vol. 90, pp. 168–183, 2017.
16. Q.-H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvag, and T.-L. ˚ Dam, “Efficient high utility itemset mining using buffered utility-lists,” Applied Intelligence, vol. 48, pp. 1859–1877, 2018.
17. P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, “A survey of itemset mining,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 7, no. 4, p. e1207, 2017.
18. J. M. Luna, P. Fournier-Viger, and S. Ventura, “Frequent itemset mining: A 25 years review,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 6, p. e1329, 2019.
19. Y. Lin, P.-C. Huang, D. Liu, X. Zhu, and L. Liang, “Making in-memory frequent pattern mining durable and energy efficient,” in 2016 45th International Conference on Parallel Processing (ICPP). IEEE, 2016, pp. 47–56.
20. Y. Lin, P.-C. Huang, D. Liu, and L. Liang, “Scalable frequent-pattern mining on nonvolatile memories,” in 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2017, pp. 578–583.
21. C. Yang, P.-C. Huang, Y. Lin, J. Dong, D. Liu, Y. Tan, and L. Liang, “Making frequent-pattern mining scalable, efficient, and compact on nonvolatile memories,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 7, pp. 1367–1380, 2020.
22. B.-E. Shie, S. Y. Philip, and V. S. Tseng, “Efficient algorithms for mining maximal high utility itemsets from data streams with different models,” Expert Systems with Applications, vol. 39, no. 17, pp. 12 947–12 960, 2012.
23. J.-F. Qu, M. Liu, and P. Fournier-Viger, “Efficient algorithms for high utility itemset mining without candidate generation,” High-Utility Pattern Mining: Theory, Algorithms and Applications, pp. 131–160, 2019.
24. C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, “Efficient tree structures for high utility pattern mining in incremental databases,” IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 12, pp. 1708–1721, 2009.
25. U. Yun and H. Ryang, “Incremental high utility pattern mining with static and dynamic databases,” Applied intelligence, vol. 42, pp. 323– 352, 2015.
26. J. Lee, U. Yun, G. Lee, and E. Yoon, “Efficient incremental high utility pattern mining based on pre-large concept,” Engineering Applications of Artificial Intelligence, vol. 72, pp. 111–123, 2018.
27. C.-J. Chu, V. S. Tseng, and T. Liang, “An efficient algorithm for mining high utility itemsets with negative item values in large databases,” Applied Mathematics and Computation, vol. 215, no. 2, pp. 767–778, 2009.
28. J. C.-W. Lin, P. Fournier-Viger, and W. Gan, “Fhn: An efficient algorithm for mining high-utility itemsets with negative unit profits,” KnowledgeBased Systems, vol. 111, pp. 283–298, 2016.